

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

2
3
4

6

7
8
9

11

12

13

15

16
17

- 19
20
21

23

24

1 claim 51, and is similarly rejected under the same rationale.” Applicant
2 respectfully submits that claim 1 is not simply a “computer-implemented
3 architecture for performing the method of claim 51.” As such, the Office has failed
4 to independently examine this claim.

5 However, in the interest of furthering prosecution of this case, Applicant
6 will attempt to respond. In its rejection of claim 51, the Office equates Applicant’s
7 “one or more objects” with Conner’s “application that was designed for statically
8 defined classes.” In the context of claim 1, Applicant believes that the Office
9 intends to equate Conner’s application with Applicant’s “one or more *first*
10 objects.” Applicant is unclear what aspect of Conner the Office would equate with
11 Applicant’s “one or more second objects” because claim 51 contains no such
12 language. Regardless, claim 1 recites “one or more second objects associated with
13 the one or more first objects and configured to *call the one or more first objects.*”
14 Applicant has studied Conner and can find nothing that would anticipate or render
15 obvious this feature. Because Conner does not disclose “one or more second
16 objects associated with the one or more first objects and configured to *call the one*
17 *or more first objects,*” it cannot disclose that the one or more first objects are
18 called to “effect one or more property value changes on the one or more first
19 objects in a manner that makes the one or more first objects appear as if they
20 support dynamic properties.” Accordingly, for at least this reason, this claim is
21 allowable.

22 **Claims 2-15** depend from claim 1 and are allowable as depending from an
23 allowable base claim. These claims are also allowable for their own recited
24 features which, in combination with those recited in claim 1, are neither disclosed
25 nor suggested in the references of record, either singly or in combination with one

1 another. Given the allowability of these claims, the rejection of claim 15 over the
2 combination with MacKay is not seen to add anything of significance.

3 4 Claims 16-23

5 **Claim 16** recites a multi-media project editing architecture comprising
6 [emphasis added]:

- 7 • one or more first objects that support only static properties, the one
8 or more first objects being configured to implement a transform
associated with processing of a multi-media editing project;
- 9 • one or more second objects associated with the one or more first
10 objects and configured to *call the one or more first objects* to effect
11 one or more property value changes on the one or more first objects
in a manner that makes the one or more first objects appear as if they
support dynamic properties; and
- 12 • one or more data structures associated with the one or more second
13 objects, individual data structures containing property data that is to
14 be used by the one or more second objects to effect a property value
change.

15 In making out the rejection of this claim, the Office incorporates the
16 rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's
17 "one or more objects" with Conner's "application that was designed for statically
18 defined classes." In the context of claim 16, Applicant believes that the Office
19 intends to equate Conner's application with Applicant's "one or more *first*
20 objects." Applicant is unclear what aspect of Conner the Office would equate with
21 Applicant's "one or more second objects" because claim 51 contains no such
22 language. Regardless, claim 16 recites "one or more second objects associated
23 with the one or more first objects and configured to *call the one or more first*
24 *objects.*" Applicant has studied Conner and can find nothing that would anticipate
25

1 or render obvious this feature. Because Conner does not disclose “one or more
2 second objects associated with the one or more first objects and configured to *call*
3 *the one or more first objects*,” it cannot disclose that the one or more first objects
4 are called to “effect one or more property value changes on the one or more first
5 objects in a manner that makes the one or more first objects appear as if they
6 support dynamic properties.” In addition, MacKay does not supply the missing
7 feature. Accordingly, for at least this reason, this claim is allowable.

8 **Claims 17-23** depend from claim 16 and are allowable as depending from
9 an allowable base claim. These claims are also allowable for their own recited
10 features which, in combination with those recited in claim 16, are neither disclosed
11 nor suggested by the references of record, either singly or in combination with one
12 another.

13 14 **Claims 24-30**

15 **Claim 24** recites a multi-media project editing architecture comprising
16 [emphasis added]:

- 17 • a software-implemented matrix switch having multiple input pins
18 and multiple output pins, the multiple input pins being routable to
19 the multiple output pins, the switch being configured to provide a
20 data stream that represents a multi-media project;
- 21 • a data structure associated with the matrix switch and configured for
22 use in programming the matrix switch to provide a routing scheme
23 for routing input pins to output pins;
- 24 • one or more first objects associated with the matrix switch, the one
25 or more first objects supporting only static properties associated with
 rendering of a multi-media project;
- one or more second objects associated with the one or more first
 objects and configured to *call the one or more first objects* to effect
 one or more property value changes on the one or more first objects

1 in a manner that makes the one or more first objects appear as if they
2 support dynamic properties.

3 In making out the rejection of this claim, the Office incorporates the
4 rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's
5 "one or more objects" with Conner's "application that was designed for statically
6 defined classes." In the context of claim 24, Applicant believes that the Office
7 intends to equate Conner's application with Applicant's "one or more *first*
8 objects." Applicant is unclear what aspect of Conner the Office would equate with
9 Applicant's "one or more second objects" because claim 51 contains no such
10 language. Regardless, claim 24 recites "one or more second objects associated
11 with the one or more first objects and configured to *call the one or more first*
12 *objects*." Applicant has studied Conner and can find nothing that would anticipate
13 or render obvious this feature. Because Conner does not disclose "one or more
14 second objects associated with the one or more first objects and configured to *call*
15 *the one or more first objects*," it cannot disclose that the one or more first objects
16 are called to "effect one or more property value changes on the one or more first
17 objects in a manner that makes the one or more first objects appear as if they
18 support dynamic properties." In addition, MacKay does not supply the missing
19 feature. Accordingly, for at least this reason, this claim is allowable.

20 **Claims 25-30** depend from claim 24 and are allowable as depending from
21 an allowable base claim. These claims are also allowable for their own recited
22 features which, in combination with those recited in claim 24, are neither disclosed
23 nor suggested by the references of record, either singly or in combination with one
24 another.
25

1 **Claims 31-40**

2 **Claim 31** recites a property value-changing method comprising [emphasis
3 added]:

- 4 • providing one or more objects that support only static properties;
- 5 • providing one or more *programmable objects* configured to effect
6 property value changes on the objects that support only static
7 properties; and
- 8 • effecting at least one property value change on the one or more
9 objects that support only static properties *using the one or more*
10 *programmable objects*.

11 In making out the rejection of this claim, the Office states that claim 31
12 “includes the same subject matter as in claim 51, and is similarly rejected under
13 the same rationale.” After linking the rejection of claim 31 to the rejection of
14 claim 51, the Office then argues that Conner discloses effecting at least one
15 property value change on the one or more objects that support only static
16 properties using the one of more programmable objects. The Office cites to
17 column 2, lines 22-43, and column 31, lines 5-17, which are reproduced below:

18 The redispatch stub invokes the objects dispatch method passing it
19 enough information to determine the correct method procedure in a
20 dynamic manner. The dispatch function then invokes the appropriate
21 method procedure and returns any return value to the redispatch stub
22 which returns it to the original application. Thus, the original static
23 method call can be converted to a dynamic call (a dispatch function
24 call) without any change on the part of the calling application's
25 binary image.

26 Redispatch stubs can be used to allow a class with a definition that
27 can vary at runtime to be used by an application that was designed
28 for statically defined classes. This is done by replacing all the entries
29 in the method procedure table for the dynamic class with the
30 appropriate redispatch stub entries during initialization of the class
31 object. Then, when the class includes suitable definitions for its
32 dispatch functions, it can change the association of methods to

1 method procedures at runtime without corrupting the execution of an
2 application using the class. *Col. 2, lines 22-43.*

3 Logic for providing a dynamic insertion of a replacement parent
4 class, referred to in object programming as a parent class shadow, is
5 detailed in this section of the invention. This processing allows the
6 statically compiled definition of what parent class is linked to a
7 particular class at runtime to be dynamically altered during
8 execution. The ability to insert a new parent class into a statically
9 compiled class hierarchy offers more flexibility to maintain and
10 enhance existing code after it has appeared in binary form. It also
11 offers a new degree of freedom for customizing code without access
12 to source materials since this result can be achieved without
13 recompilation. *Col. 31, lines 5-17.*

14 In its rejection of claim 51, the Office equates Applicant's "one or more
15 objects that support only static properties" with Conner's "application that was
16 designed for statically defined classes." Applicant is unclear what aspect of
17 Conner the Office would equate with Applicant's "programmable object" because
18 claim 51 contains no such language. Regardless, Applicant can find nothing in the
19 cited excerpts, or anywhere else in Conner, that would disclose or suggest
20 "providing one or more *programmable objects* configured to effect property value
21 changes on the objects that support only static properties" as that term is used in
22 the context of Applicant's specification. Because Conner does not disclose or
23 suggest providing one or more programmable objects, as the Applicant uses the
24 term, it cannot disclose or suggest "effecting at least one property value change on
25 the one or more objects that support only static properties *using the one of more
programmable objects.*" Accordingly, for at least this reason, this claim is
allowable.

Claims 32-40 depend from claim 31 and are allowable as depending from
an allowable base claim. These claims are also allowable for their own recited

1 features which, in combination with those recited in claim 31, are neither disclosed
2 nor suggested by the references of record, either singly or in combination with one
3 another.

4 5 **Claims 41-44**

6 **Claim 41** recites a property value-changing method comprising [emphasis
7 added]:

- 8 • programming a programmable object with property data that defines
9 when certain property value changes are to be made and what those
10 property value changes are;
- 11 • *calling, with the programmable object, one or more objects that do*
12 *not support dynamic properties*; and
- 13 • responsive to said calling, using the property data to effect a
14 property value change on the one or more objects that do not support
15 dynamic properties.

16 In making out the rejection of this claim, the Office states that claim 41
17 “includes the same subject matter as in claim 31, and is similarly rejected under
18 the same rationale.” The rejection of claim 31 states that “claim 31 includes the
19 same subject matter as in claim 51, and is rejected under the same rationale.”
20 Applicant respectfully submits that each of these claims (claims 31, 41, and 51)
21 merits its own independent examination, but Applicant will try to follow the
22 Office’s logic.

23 In its rejection of claim 51, the Office equates Applicant’s “one or more
24 objects that support only static properties” with Conner’s “application that was
25 designed for statically defined classes.” In the context of claim 41, Applicant
believes that the Office intends to equate Conner’s application with Applicant’s
“one or more objects that do not support dynamic properties.” Applicant is unclear

1 what aspect of Conner the Office would equate with Applicant's "programmable
2 object" because claim 51 contains no such language. Regardless, claim 41 recites
3 *"calling, with the programmable object, one or more objects that do not support*
4 *dynamic properties."* Applicant has studied Conner and can find nothing that
5 would anticipate or render obvious this feature. Because Conner does not disclose
6 "calling, with the programmable object, one or more objects that do not support
7 dynamic properties," it cannot disclose, responsive to the act of calling, "using the
8 property data to effect a property value change on the one or more objects that do
9 not support dynamic properties." Accordingly, for at least this reason, this claim is
10 allowable.

11 **Claims 42-44** depend from claim 41 and are allowable as depending from
12 an allowable base claim. These claims are also allowable for their own recited
13 features which, in combination with those recited in claim 41, are neither disclosed
14 nor suggested by the references of record, either singly or in combination with one
15 another.

16 17 **Claim 45**

18 **Claim 45** recites one or more computer-readable media having computer-
19 readable instructions thereon which, when executed by a computer, cause the
20 computer to [emphasis added]:

- 21
- 22 • provide one or more objects that support only static properties;
 - 23 • provide one or more programmable objects configured to effect
24 property value changes on the objects that support only static
25 properties;
- program the one or more programmable objects with property data
that is to be used by the one or more programmable objects to effect
said at least one property value change, the property data

comprising: property value changes that are to be made, time(s) at which property value changes are to be made, and how the property value changes are to be made; and effect at least one property value change on the one or more objects that support only static properties *by using the one or more programmable objects to call the one or more objects that support only static properties.*

In making out the rejection of this claim, the Office states that claim 45 “includes the same subject matter as in claim 31, and is similarly rejected under the same rationale.” The rejection of claim 31, in turn, states that “claim 31 includes the same subject matter as in claim 51, and is similarly rejected under the same rationale.” After linking the rejection of claim 45 to the rejections of claims 31 and 51, the Office then argues that Conner discloses effecting at least one property value change on the one or more objects that support only static properties by using the one of more programmable objects to call the one or more objects that support only static properties. The Office cites to column 2, lines 32-43, and column 31, lines 5-17, which are reproduced below:

Redispatch stubs can be used to allow a class with a definition that can vary at runtime to be used by an application that was designed for statically defined classes. This is done by replacing all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries during initialization of the class object. Then, when the class includes suitable definitions for its dispatch functions, it can change the association of methods to method procedures at runtime without corrupting the execution of an application using the class. *Col. 2, lines 32-43.*

Logic for providing a dynamic insertion of a replacement parent class, referred to in object programming as a parent class shadow, is detailed in this section of the invention. This processing allows the statically compiled definition of what parent class is linked to a particular class at runtime to be dynamically altered during execution. The ability to insert a new parent class into a statically compiled class hierarchy offers more flexibility to maintain and

1 enhance existing code after it has appeared in binary form. It also
2 offers a new degree of freedom for customizing code without access
3 to source materials since this result can be achieved without
4 recompilation. *Col. 31, lines 5-17.*

5 In its rejection of claim 51, the Office equates Applicant's "one or more
6 objects that support only static properties" with Conner's "application that was
7 designed for statically defined classes." Applicant is unclear what aspect of
8 Conner the Office would equate with Applicant's "programmable object" because
9 claim 51 contains no such language. Regardless, Applicant can find nothing in the
10 cited excerpts, or anywhere else in Conner, that would disclose or suggest
11 effecting "at least one property value change on the one or more objects that
12 support only static properties *by using the one of more programmable objects to*
13 *call the one or more objects that support only static properties.*" Accordingly,
14 for at least this reason, this claim is allowable.

15 **Claims 46-50**

16 **Claim 46** recites a property value-changing method comprising [emphasis
17 added]:

- 18 • programming a programmable object with property data that defines
19 when certain property value changes are to be made and what those
20 property value changes are, the property value changes being
21 associated with rendering of a multi-media editing project;
- 22 • *calling, with the programmable object, one or more objects that do*
23 *not support dynamic properties*; and
- 24 • responsive to said calling, using the property data to effect a
25 property value change on the one or more objects.

1 In making out the rejection of this claim, the Office states that claim 46
2 “includes the same subject matter as in claim 31, and is similarly rejected under
3 the same rationale.” The rejection of claim 31 states that “claim 31 includes the
4 same subject matter as in claim 51, and is rejected under the same rationale.”
5 Applicant respectfully submits that each of these claims (claims 31, 46, and 51)
6 merits its own independent examination, but Applicant will try to follow the
7 Office’s logic.

8 In its rejection of claim 51, the Office equates Applicant’s “one or more
9 objects that support only static properties” with Conner’s “application that was
10 designed for statically defined classes.” Applicant is unclear what aspect of
11 Conner the Office would equate with Applicant’s “programmable object” because
12 claim 51 contains no such language. Regardless, claim 46 recites “*calling, with*
13 *the programmable object, one or more objects that do not support dynamic*
14 *properties.*” Applicant has studied Conner and can find nothing that would
15 anticipate or render obvious this feature. Because Conner does not disclose
16 “calling, with the programmable object, one or more objects that do not support
17 dynamic properties,” it cannot disclose, responsive to the act of calling, “using the
18 property data to effect a property value change on the one or more objects.” In
19 addition, MacKay does not supply the missing feature. Accordingly, for at least
20 this reason, this claim is allowable.

21 **Claims 47-50** depend from claim 46 and are allowable as depending from
22 an allowable base claim. These claims are also allowable for their own recited
23 features which, in combination with those recited in claim 46, are neither disclosed
24 nor suggested by the references of record, either singly or in combination with one
25 another.

1
2 **Claims 51-54**

3 **Claim 51** recites a property value-changing method comprising [emphasis
4 added]:

- 5
- 6 • providing one or more objects that support only static properties; and
 - 7 • simulating dynamic properties with the one or more objects by
8 changing one or more property values *at a pre-programmed time*.

9 In making out the rejection of this claim, the Office argues that Conner
10 discloses simulating dynamic properties with the one or more objects by changing
11 one or more property values at a pre-programmed time. In particular, the Office
12 states that “the original static method call can be converted to a dynamic call . . .
13 this is done by replacing all the entries in the method procedure table for the
14 dynamic class with the appropriate redispatch stub entries during initialization of
15 the class object.” The Office then cites to column 2, lines 12-43, column 31, lines
16 5-17, and column 33, lines 6-22, all of which are reproduced below [emphasis
17 added]:

18 These and other objectives of the present invention are accomplished
19 by the operation of an algorithm in the memory of a processor that
20 uses redispatch method stubs to convert static method calls into
21 dynamic method calls. When the System [sic] Object Model
22 (SOM) compiler generates support for defining a class, it generates a
23 redispatch stub for each method defined in the class. A redispatch
24 stub is a short sequence of instructions that has the exact same
25 calling sequence as its corresponding method.

23 The redispatch stub invokes the objects dispatch method passing it
24 enough information to determine the correct method procedure in a
25 dynamic manner. The dispatch function then invokes the appropriate
method procedure and returns any return value to the redispatch stub
which returns it to the original application. Thus, the original static

1 method call can be converted to a dynamic call (a dispatch function
2 call) without any change on the part of the calling application's
3 binary image.

4 Redispatch stubs can be used to allow a class with a definition that
5 can vary at runtime to be used by an application that was designed
6 for statically defined classes. This is done by replacing all the entries
7 in the method procedure table for the dynamic class with the
8 appropriate redispatch stub entries *during initialization of the class*
9 *object*. Then, when the class includes suitable definitions for its
10 dispatch functions, it can change the association of methods to
11 method procedures at runtime without corrupting the execution of an
12 application using the class. *Col. 2, lines 12-43.*

13 Logic for providing a dynamic insertion of a replacement parent
14 class, referred to in object programming as a parent class shadow, is
15 detailed in this section of the invention. This processing allows the
16 statically compiled definition of what parent class is linked to a
17 particular class at runtime to be dynamically altered during
18 execution. The ability to insert a new parent class into a statically
19 compiled class hierarchy offers more flexibility to maintain and
20 enhance existing code after it has appeared in binary form. It also
21 offers a new degree of freedom for customizing code without access
22 to source materials since this result can be achieved without
23 recompilation. *Col. 31, lines 5-17.*

24 The invention consists of a programming mechanism called
25 redispatch stubs to ameliorate the difference between static and
dynamic models. A redispatch stub is a small procedure with an
entry point that can be placed into a table of procedure entry points.
The table of procedure entry points are used in a static object model
as a substitute for the actual method entry point that is expected. The
redispatch stub is generated automatically based on the requirements
of the dynamic object model. The redispatch stub converts the call
generated in the static object model into the form necessary in the
dynamic object model and supplies any missing information in the
process. Thus, if an object is accessed from a static object model that
is provided by a dynamic object model, it can be represented to the
static object model via a table of entry points which each indicate a
particular redispatch stub. *Col. 33, lines 6-22.*

1 As discussed in the excerpts cited, Conner replaces all the entries in the
2 method procedure table for the dynamic class with the appropriate redispach stub
3 entries *during initialization of the class object*. In contrast, Applicant simulates
4 dynamic properties with one or more objects that support only static properties by
5 changing one or more property values *at a pre-programmed time*. An example of
6 how this might work is given in the specification at page 54, lines 1-24,
7 reproduced below:

8 Consider the following example where object 4000 comprises a ball.
9 Consider that, for the ball, we want the helper object to cause the ball to be
10 red at time 0, change immediately to blue at time 2, and blend into yellow
11 at time 4. An exemplary data structure that can enable this to be done is
12 shown in Fig. 40. In this example, the data structure "(0, Red, Jump)"
13 indicates that at time 0, the ball is to immediately (i.e. "jump to") be red;
14 the structure "(2, Blue, Jump)" indicates that at time 2, the ball is to
15 immediately be blue; and the structure "(4, Yellow, Interpolate)" indicates
16 that at time 4, the ball is to change in a linear fashion (i.e. interpolate) into
17 yellow from blue.

18 The helper object 4004 is programmed (or pre-programmed) with the
19 various data structures that define the desired properties, their value
20 changes, and the time and manner in which the value change is to take
21 place. The helper object keeps track of the time, by for example, being
22 called or otherwise notified of the time. At the appropriate times, the helper
23 object, using the array of structures, calls the object whose property values
24 are desired to be changed so that the property values can be changed. In the
25 present example, at time 0, the helper object calls object 4000 or a property
object on the object itself, and causes the property value for the color
property to be set to "red". At time 2, the helper object 4004 calls the
object 4000 and causes the property value for the color property to be set to
"blue". At time 4, the helper object calls the object 4000 and causes the
property value for the color property to interpolate from its previous value
(i.e. blue) to the present value, i.e. yellow.

Thus, object 4000' appears as if it supports dynamic properties when, in
fact, it only supports static properties.

1 Thus, Conner neither discloses nor suggests simulating dynamic properties
2 with one or more objects that support only static properties by changing one or
3 more property values *at a pre-programmed time*. Accordingly, for at least this
4 reason, this claim is allowable.

5 **Claims 52-54** depend from claim 51 and are allowable as depending from
6 an allowable base claim. These claims are also allowable for their own recited
7 features which, in combination with those recited in claim 51, are neither disclosed
8 nor suggested by the references of record, either singly or in combination with one
9 another. Given the allowability of these claims, the rejection of claim 54 over the
10 combination with MacKay is not seen to add anything of significance.

11 12 **Claims 55-58**

13 **Claim 55** recites a multi-media system comprising [emphasis added]:

- 14 • an application program configured to enable a user to define a multi-
15 media project in which multiple digital source streams can be
combined;
- 16 • a software-implemented matrix switch having multiple input pins
17 and multiple output pins, the input pins being individually associated
18 with inputs that can compete, during a common time period, for a
particular output pin that is associated with the matrix switch, the
19 switch being configured to receive, at its input pins, digital source
streams;
- 20 • a first data structure associated with the matrix switch and
21 configured for use in programming the matrix switch to provide a
routing scheme for routing input pins to output pins such that at any
22 given time, only one input pin is routed to the particular output pin;
- 23 • a second data structure associated with and different from the first
24 data structure, the second data structure representing a user-defined
25 multi-media project and being configured so that the first data
structure can be derived therefrom;

1 one or more first objects associated with the matrix switch, the one
2 or more first objects supporting only static properties associated with
3 rendering of a multi-media project; and

- 4 • one or more second objects associated with the one or more first
5 objects and configured to ***call the one or more first objects*** to effect
6 one or more property value changes on the one or more first objects
7 in a manner that makes the one or more first objects appear as if they
8 support dynamic properties.

9 In making out the rejection of this claim, the Office incorporates the
10 rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's
11 "one or more objects" with Conner's "application that was designed for statically
12 defined classes." In the context of claim 55, Applicant believes that the Office
13 intends to equate Conner's application with Applicant's "one or more ***first***
14 ***objects.***" Applicant is unclear what aspect of Conner the Office would equate with
15 Applicant's "one or more second objects" because claim 51 contains no such
16 language. Regardless, claim 55 recites "one or more second objects associated
17 with the one or more first objects and configured to ***call the one or more first***
18 ***objects.***" Applicant has studied Conner and can find nothing that would anticipate
19 or render obvious this feature. Because Conner does not disclose "one or more
20 second objects associated with the one or more first objects and configured to ***call***
21 ***the one or more first objects,***" it cannot disclose that the one or more first objects
22 are called to "effect one or more property value changes on the one or more first
23 objects in a manner that makes the one or more first objects appear as if they
24 support dynamic properties." In addition, MacKay does not supply the missing
25 feature. Accordingly, for at least this reason, this claim is allowable.

Claims 56-58 depend from claim 55 and are allowable as depending from
an allowable base claim. These claims are also allowable for their own recited

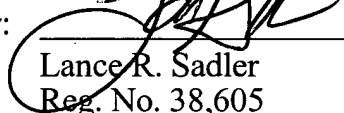
1 features which, in combination with those recited in claim 55, are neither disclosed
2 nor suggested by the references of record, either singly or in combination with one
3 another.

4
5 **Conclusion**

6 Applicant submits that all of the claims are in condition for allowance and
7 respectfully requests that the Office pass the application along to issuance. If the
8 Office's next anticipated action is to be anything other than issuance of a Notice of
9 Allowability, Applicant respectfully requests a telephone call for the purpose of
10 scheduling an interview.

11
12 Respectfully Submitted,

13
14 Dated: 7/8/04

15 By: 
16 Lance R. Sadler
17 Reg. No. 38,605
18 (509) 324-9256
19
20
21
22
23
24
25